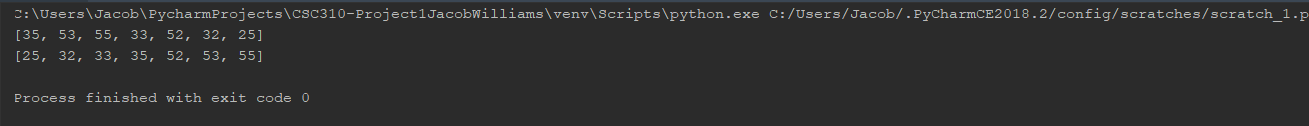
CSC310 Project 1 Description

By Jacob W

The first part of the assignment was to implement a Radix sort using a Queue. Below you will find the screenshot of the code as well as the results of running it.

def radixSort(a):  
 # Creates several buckets in which to store the values while being sorted at each digit  
 arrOfQs = [ArrayQueue(), ArrayQueue(), ArrayQueue(), ArrayQueue(), ArrayQueue(), ArrayQueue(), ArrayQueue(),  
 ArrayQueue(), ArrayQueue(), ArrayQueue()]  
  
 max = a[0]; # Finds the max value of array  
 for i in range(1, len(a)):  
 if (a[i] > max):  
 max = a[i];  
  
 count = 0 # Finds the highes amount of digits (10^i) of the max number in array  
 while max > 0:  
  
 if max == 0:  
 count += 1  
 count += 1  
 # Integer divide by ten in order to "count" each digit  
 max = max // 10  
  
 # For each value from zero to the amount of digits of our highest value  
 for i in range(1, count + 1):  
 cnt = 0; # -Keeps track of how many are currently in "buckets"  
 # For each item in array  
 for j in range(len(a)):  
 # Set the bucket whose index is equal to the value of the digit in question to the value of item in the array  
 arrOfQs[a[j] % (10 \*\* i) // 10 \*\* (i - 1)].enqueue(a[j])  
 # For each bucket  
 for j in range(len(arrOfQs)):  
 # While there are values in the bucket  
 while (not arrOfQs[j].is\_empty()):  
 # Cycle pur each value back into the array  
 a[cnt] = arrOfQs[j].dequeue()  
 # Note which value we are examining  
 cnt += 1;

Editor:



I first approached this problem knowing that to properly implement a radix sort, certain things were required. For one, I knew I was going to need ten different “buckets” to store the values while their digits were being sorted, I also knew I was required to use ArrayQueues for this assignment so that seemed to be a very logical choice. I also knew I was going to need to sort the values of each digit for each item in the original array. This of course meant I was going to need to figure out what the highest number of digits I was going to need to sort. This also required me to know *which* value in my array was the highest. Both problems were trivially handled by implementing two simple intuitive loops. The first (designed to find the max value) simple iterated through the array until it found the highest values. Then I simply integer divided that value by ten until it was zero (keeping track of how many times I done so). Thus, giving me the highest number of digits. After that I implemented a for loop which (for each digit my highest value has) would first, assign each item in to an ArrayQueue “bucket” whose index was the same as the digit in question, then “empty” those ArrayQueues back into my array. This meant that each digit was sorted by first putting them into assigned ArrayQueues and then emptying in order. After the main loop ends the array is sorted.

The second portion of the assignment was to implement a method which would evaluate an expression written in Postfix notation. See below for screenshots of the code as well as the results.

def postFixConvert(e):  
 numStack = ArrayStack();  
  
 for i in range(len(e)): # For every char in the string  
 if e[i] in ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]: # If the char in question is a number  
 numStack.push(int(e[i])) # Add it to the stack  
 else: # Otherwise (if the value is an operator)  
  
 val1 = numStack.pop() # Remove the digits it from the stack and store them  
 val2 = numStack.pop() # (This is also so we can subtract and divide without worrying about the order)  
 if e[i] == "+": # If the operator is 'Add'  
 numStack.push(val2 + val1) # Push the sum of the two values to the stack  
 if e[i] == "−": # If the operator is 'Subtract'  
 numStack.push(val2 - val1) # Push the difference to the stack  
 if (e[i] == "∗"): # If the operator is 'Multiply'  
 numStack.push(val2 \* val1) # Push the product to the stack  
 if e[i] == "/": # If the operator is 'Divide'  
 numStack.push(val2 / val1) # Push the 'Quotient' to the stack  
  
 # This ensures that every two values before an operator condensed into one, ensuring we will only have one value at the end  
 return numStack.pop() # return only value in the stack

Editor:



When I first approached this problem, I knew I was going to need to loop through a string and I distinguish between a numerical character and an operator character. So, I first made a for loop which would iterate over the string while making the necessary comparisons. My first comparison was simply to see if the character in question a number was, this was done by comparing the character to an array of numbers for matches. If it was the same, I simply added the number to the stack. If it wasn’t however I knew I had to make an operation. I also knew that some operations required the first value written to be evaluated first (e.g. subtraction which requires the second number to be removed from the first). This doesn’t work with stack which only provided access to the most recently added value. To get around the problem I assigned two variables in order to “switch” the position of the two numbers in the stack. This was helpful in that it allowed me to simplify division and subtraction without affect addition and multiplication. After the variables were assigned I simply checked which operating symbol was equal to e[i] and did the appropriate operation. I also pushed the result of the operation to the stack (allowing me to perform my next operation on it) after I had iterated the entire string I simply return the “pop” of the only value in the stack.

Note: In this assignment I used the ArrayQueue and ArrayStack classes provided in class.